

ADANA SCIENCE AND TECHNOLOGY UNIVERSITY

Introduction to Computer Programming II

STRING CLASS

String

- String is a collection of characters.
- There are two types of strings commonly used in C++ programming language:
  - Strings that are objects of string class (The Standard C++ Library string class)
  - C-strings (C-style Strings)

C-strings

- In C programming, the collection of characters is stored in the form of arrays, this is also supported in C++ programming.
- Hence it's called C-strings.
- C-strings are *arrays of type char* <u>terminated</u> with null character, that is, \0 (ASCII value of null character is 0).
- To define a C-string
  - char str[] = "C++";
- In the above code, str is a string and it holds 4 characters.
- Although, "C++" has 3 character, the null character \0 is added to the end of the string automatically.

Alternative ways of defining a string

- char str[4] = "C++";
- char str[] = {'C','+','+','\0'};
- char str[4] = {'C','+','+','\0'};
- Like arrays, it is not necessary to use all the space allocated for the string.
- For example:
  - char str[100] = "C++";

Example

```
#include <iostream>
using namespace std;

int main()
{
char str[100];

cout << "Enter a string: ";
cin >> str;
cout << "You entered: " << str
        << endl;

cout << "\nEnter another string: ";
cin >> str;
cout << "You entered: "<<str<<endl;

return 0;
}
```

Enter a string: C++
You entered: C++

Enter another string: Programming is fun.
You entered: Programming

- Notice that, in the second example only "Programming" is displayed instead of "Programming is fun"
- This is because the extraction operator >> works as scanf() in C and considers a space " " has a terminating character.

C String to read a line of text

```
#include <iostream>
using namespace std;

int main()
{
char str[100];
cout << "Enter a string: ";
cin.get(str, 100);

cout << "You entered: " << str
        << endl;
return 0;
}
```

- Enter a string:
  Programming is fun.
- You entered:
  Programming is fun.

C String to read a line of text

- To read the text containing blank space, **cin.get** function can be used.
- This function takes two arguments.
  - `cin.get(str, 100);`
- First argument is the name of the string (address of first element of string) and second argument is the maximum size of the array.
  - Remember: `char str[100];`
- In the above program, str is the name of the string and 100 is the maximum size of the array.

string Object

- In C++, you can also create a string object for holding strings.
- Unlike using char arrays, string objects has no fixed length, and can be extended as per your requirement.

C++ string using string data type

```
#include <iostream>
using namespace std;

int main()
{
// Declaring a string object
string str;
cout << "Enter a string: ";
getline(cin, str);

cout << "You entered: " << str
        << endl;
return 0;
}
```

- Enter a string: Programming is fun.
- You entered: Programming is fun.

C++ string using string data type

- In this program, a string str is declared.
- Then value for the string is asked from the user.
- Instead of using cin>> or cin.get() function, you can get the entered line of text using getline().
- getline() function takes the input stream as the first parameter which is cin and str as the location of the line to be stored.

C++ string

- C++ string class internally uses char array to store character
- but all memory management, allocation and null termination is handled by string class itself that is why it is easy to use
- The length of C++ string can be changed at runtime because of dynamic allocation of memory.
- C++ string class has a lot of functions to handle string easily.

String declaration

- There are various declation styles for strings:

```
 // various constructors of string class
 // initialization by raw string
string str1("first string");

// initialization by another string
string str2(str1);

// initialization by character with number of occurrence
string str3(5, '#');
//str is equal to #####

// initialization by part of another string
string str4(str1, 6, 3); // from 6th index (second parameter)
// 3 characters (third parameter).
//str4 : str  (str4 would be string if 3rd parameter was missing)

// initialization by part of another string : iterator version
string str5(str2.begin(), str2.begin() + 3);
//str5: fir
```

String functions

```
string str1;
string str2="to be or not to be-that is the question";

str1.assign(str2);
//to be or not to be-that is the question

cout<<str1<<endl;
str1.assign(str2,6,11);
//or not to b

str1.assign("What is the REAL problem?",4);
//What

str1.assign(5,'*');
//*****

str1.assign(5,0x2D);
//-----

str1.assign(str2.begin()+6,str2.end()-11);
//or not to be-that is t
```

String functions

```
// assignment operator
string str6 = str4;

// clear function deletes all characters from string
 str4.clear();

// both size() and length() return length of string and
// they work as synonyms
 int len = str6.length(); // Same as "len = str6.size();"

 cout << "Length of string is : " << len << endl;
 //Length of string is : 3 (remember : str6 = "str")

// a particular character can be accessed using at or [] operator
 char ch = str6.at(2); // Same as "ch = str6[2];"
 cout << "third character of string is : " << ch << endl;
```

String functions

```
string str6= "string";
// append add the argument string at the end
str6.append(" extension");
// same as str6 += " extension"
// str6: string extension

// another version of append, which appends part of other string
str4 ="fly";
str4.append(str6,3,3);
// str4 : flying
```

String functions

```
// find returns index where pattern is found.
// If pattern is not there it returns predefined constant npos
// whose value is -1

if (str6.find(str4) != string::npos)
  cout << "str4 found in str6 at " << str6.find(str4)
       << " pos" << endl;
else
  cout << "str4 not found in str6" << endl;
```

npos is a static member constant value with the greatest possible value for an element of type size_t.

This value, when used as the value for a *len* parameter in string's member functions, means *"until the end of the string"*.

String functions

- For example:

```
string str6= "string";
str4 ="ing";
cout<<str6.find(str4)<<endl;
```

  - will print 3

```
string str6= "string";
str4 ="fly";
cout<<str6.find(str4)<<endl;
```

  - will print something like 4294967295
  - Indicating that str4 is NOT found in str6
  - Can be controlled using string::npos
  - `str6.find(str4) != string::npos`
    - is TRUE if str4 is in str6
    - FALSE otherwise

String functions

```
// substr(a, b) function returns a substring of length b
// starting from index a
cout << str6.substr(7, 3) << endl;
// str6 : string extension
// output : ext
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s	t	r	i	n	g		e	x	t	e	n	s	i	o	n

```
// if second argument is not passed, string till end is
// taken as substring
cout << str6.substr(7) << endl;
// output : extension
```

String functions

```
// erase(a, b) deletes b character at index a
// str6 : string extension
str6.erase(7, 4);
// str6 : string nsion
cout << str6 << endl;
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s	t	r	i	n	g		**e**	**x**	**t**	**e**	n	s	i	o	n

0	1	2	3	4	5	6	7	8	9	10	11
s	t	r	i	n	g		n	s	i	o	n

```
// iterator version of erase
str6.erase(str6.begin() + 5, str6.end() - 3);
// str6 : strinion
cout << str6 << endl;
```

0	1	2	3	4	5	6	7	8	9	10	11
s	t	r	i	**n**	**g**		**n**	**s**	i	o	n

0	1	2	3	4	5	6
s	t	r	i	i	o	n

Operators

```
string str7 = "Beautiful ";
cout<<str7+"weather today"<<endl;
// str7 : Beautiful
//output : Beautiful weather today

str7+="weather today";
// str7 : Beautiful weather today
```

insert Function

Inserts additional characters into the string right before the character indicated

```
string str1="This is a sentence";
string str2="Franny";
string str3=" nice";
str1.insert(9,str3);
cout<<str1<<endl;
str1.insert(9,str2,4,2);
cout<<str1<<endl;
str1.insert(20,3,'!');
cout<<str1<<endl;
str1.insert(9," very well equipped",6);
cout<<str1<<endl;
str1.insert(9,"good",1,2);
cout<<str1<<endl;
```

```
This is a nice sentence
This is any nice sentence
This is any nice sen!!!tence
This is a very ny nice sen!!!tence
This is aoo very ny nice sen!!!tence
```

replace Function

- Replaces the portion of the string

```
string str1="This is a test sentence.";
string str2="fantastic";
string str3="nice string block";
str1.replace(10,4,str2);
cout<<str1<<endl;
cout<<"len:"<<str1.length()
<<endl;
str1.replace(20,8,str3,5,6);
cout<<str1<<endl;
str1.replace(8,1,"an extremely useful",9);
cout<<str1<<endl;
str1.replace(str1.length()-1,1,5,'!');
cout<<str1<<endl;
```

```
This is a fantastic sentence.
len:29
This is a fantastic string.
This is an extrem fantastic string.
This is an extrem fantastic string!!!!!
```

swap Function

- Exchanges the content of the container by the content of *str*, which is another string object. Lengths may differ.

```
string name1="Ali";
string name2="Veli";
cout<<name1<<" - «
     <<name2<<endl;
name1.swap(name2);
cout<<name1<<" - «
     <<name2<<endl;
```

```
Ali - Veli
Veli - Ali
```

substr Function

- Returns a newly constructed string object with its value initialized to a copy of a substring of this object.

```
string str="We think in generalities, but we live in details.";
// (quoting Alfred N. Whitehead)
string str2 = str.substr (3,5);
// "think"
cout<<str2<<endl;
size_t pos = str.find("live");
// position of "live" in str
cout<<"pos : "<<pos<<endl;
string str3 = str.substr (pos);
// get from "live" to the end
cout << str2 << ' ' << str3 << '\n';
```

```
think
pos : 33
think live in details.
```

find Function

- Searches the string for the first occurrence of the sequence specified by its arguments.

```
string str="to be or not to be-that is the question.";
string str2 ="to be";
size_t found;
found=str.find(str2);
if(found!=string::npos)
    cout<<"First occurence of "<<str2<<" is at "<<int(found)<<endl;
found = str.find(str2,found+1);
cout<<"Second occurence of "<<str2<<" is at "<<int(found)<<endl;
found = str.find('.');
cout<<"Sentence ends at "<<int(found)<<endl;
```

```
First occurence of to be is at 0
Second occurence of to be is at 13
Sentence ends at 39
```

find Function

```
//change the first occurence of to be to good
str="to be or not to be-that is the question.";
cout<<"Going into while loop.."<<endl;
//to change all occurences
while(true)
{
    found=str.find(str2);
    if(found==string::npos)
        break;
    str.replace(found,str2.length(),"good");
    cout<<str<<endl;
}
```

```
good or not to be-that is the question.
Going into while loop..
good or not to be-that is the question.
good or not good-that is the question.
```

rfind Function

- Searches the string for the last occurrence of the sequence specified by its arguments.

```
string str="to be or not to be...";
string str2 ="to be";
size_t found;

//to change all occurences
while(true)
{
    found=str.rfind(str2);
    if(found==string::npos)
        break;
    str.replace(found,str2.length(),"good");
    cout<<str<<endl;
}
```

```
to be or not good...
good or not good...
```

find_first_of/find_last_of

- Searches the string for the first/last character that matches **any** of the characters specified in its arguments.

```
string str="search the chair.";
size_t found1;
size_t found2;

found1=str.find_first_of("abcde");
found2=str.find_last_of("abcde");
cout<<"found1 : "<<found1
    <<" ch: "<<str[found1]<<endl;
cout<<"found2 : "<<found2
    <<" ch: "<<str[found2]<<endl;
```

```
found1 : 1 ch: e
found2 : 13 ch: a
```

find_first_not_of/find_last_not_of

- Searches the string for the first/last character that matches **any** of the characters specified in its arguments.

```
string str="search the chair.";
size_t found1;
size_t found2;

found1=str.find_first_not_of("abcde");
found2=str.find_last_not_of("abcde");
cout<<"found1 : "<<found1
    <<" ch: "<<str[found1]<<endl;
cout<<"found2 : "<<found2
    <<" ch: "<<str[found2]<<endl;
```

```
found1 : 0 ch: s
found2 : 18 ch: .
```

compare Function

- Compares the value of the string object (or a substring) to the sequence of characters specified by its arguments.

```
string str1 ("green apple");
string str2 ("red apple");

if (str1.compare(str2) != 0)
    cout << str1 << " is not " << str2 << '\n';

if (str1.compare(6,5,"apple") == 0)
    cout << "still, " << str1 << " is an apple\n";

if (str2.compare(str2.size()-5,5,"apple") == 0)
    cout << "and " << str2 << " is also an apple\n";

if (str1.compare(6,5,str2,4,5) == 0)
    cout << "therefore, both are apples\n";
```

```
green apple is not red apple
still, green apple is an apple
and red apple is also an apple
therefore, both are apples
```

compare Function

```
string str1 ("green apple");
string str2 ("red apple");
//What is the compare result
cout<<str1.compare(str2)<<endl;
//-1 since g<r
cout<<str2.compare(str1)<<endl;
//1 since r>g
cout<<str1.compare(2,1,str2,2,1)<<endl;
//Compare e to d. result is 1 since e>d
cout<<str1.compare(2,1,str2,1,1)<<endl;
//Compare e to e. result is 0 since e==e
cout<<str1.compare(2,3,str2,1,1)<<endl;
//Compare een to e. result is 2 since e==e
//But length of een is longer than e by 2
```


Q&A